To build an **NLP pipeline** that extracts finance-related tweets (e.g., #nifty50, #sensex, #banknifty, #intraday), cleans and analyzes them, and generates **daily sentiment-based trading signals** using keyword heuristics and embeddings.

Overview of Pipeline

1. **Tweet Scraping (Playwright)**
2. **Data Cleaning and Normalization**
3. **Feature Engineering (TF-IDF, embeddings, keyword scoring)**
4. **Signal Aggregation & Classification**
5. **Visualization (PCA)**
6. **Streamlit App (1-click interface)**

---

1. Data Collection

- **Tool:** Playwright (headless browser automation)
- **Authentication:** Uses a logged-in Twitter session via twitter_storage.json
- **Search Queries:** #nifty50, #banknifty, #sensex, #intraday
- **Actions:**
    - Scrolls through the Twitter search result page
    - Extracts:
        - username
        - timestamp
        - content
        - likes, retweets
        - mentions, hashtags
- **Output:** Saved as all_tags_tweets.parquet

Anti-bot Handling

- Mimics scrolling with randomized delay

- Bypasses login by using session state

---

2. Data Cleaning

- **Script:** clean_tweets.py
- **Steps:**
    - Normalize Unicode
    - Remove emojis and special characters
    - Strip extra whitespace, newlines
    - Parse & localize timestamps
    - Deduplicate tweets using (username, timestamp, content)
    - Language detection (English/Hindi only)
- **Output:** tweets_cleaned.parquet

---

3. ⚙️ Feature Engineering

- **TF-IDF Vectors:**
    - Converts cleaned text into sparse vectors
    - Saved as tfidf_vectors.npz + tfidf_vectorizer.pkl
- **Sentence Embeddings:**
    - Model: all-MiniLM-L6-v2 via sentence-transformers
    - Stored in tweets_with_embeddings.parquet
- **Custom Features:**
    - Domain-specific keyword score (buy/sell weightage)
    - Example: "buy", "breakout" → +1; "bearish", "fall" → −1
- **Output:** tweets_with_keywordscore.parquet

---

4. 📈 Signal Aggregation

- **Script:** aggregate_signals.py
- **Daily Grouping by Date**
- **Calculates:**
    - Volume of tweets
    - Percentage of buy, sell, neutral
    - Average keyword score
    - Composite score (weighted):

CopyEdit

0.5 * buy_pct + 0.3 * keyword_score + 0.2 * volume_factor

- **Thresholds:**
    - > 0.65 → "buy"
    - < 0.35 → "sell"
    - else → "neutral"
- **Output:** daily_aggregated_signals.parquet

---

5. 📊 Visualization

- **Script:** visualize.py
- **Plots:**
    - **PCA scatter plot** of:
        - TF-IDF vectors
        - Sentence embeddings
- **Saved to:** visualizations/

---

6. 🌐 Streamlit App

- **Script:** app.py
- **Usage:** streamlit run app.py

- **Can be deployed to Streamlit cloud**

📦 File Outputs

File	Description
all_tags_tweets.parquet	Raw scraped tweets
tweets_cleaned.parquet	Cleaned and filtered tweets
tfidf_vectors.npz	TF-IDF vector matrix
tweets_with_embeddings.parquet	Sentence BERT vectors
tweets_with_keywordscore.parquet	Includes custom keyword signal
daily_aggregated_signals.parquet	Final trading signal with confidence score
tweet_wordcloud.png	Word cloud of tweet content
top_hashtags.png	Hashtag frequency chart

Key Highlights

- Unicode normalization and emoji handling for Indian languages
- Heuristic + NLP-based sentiment aggregation
- Scalable and modular Python pipeline
- Deployable as a Streamlit app
- Memory-efficient: sparse matrices, sampled visualizations

Drawbacks

- The tweet metrics scraping does not give correct results everytime.
- Word cloud and hashtag visualization can be added
- The scrapping logic can break if twitter UI gets modified .

Future Enhancements

- Add Twitter API fallback support (if scraping blocked)
- Integrate LLM sentiment scoring (e.g., via OpenAI/GPT)
- Real-time dashboard with refreshable signals
- Backtesting with market data